D0 Trigger Database Taskforce
Plans for Evolution of Trigger Database after February 28, 2006

Compiled by Jim Linnemann, Igor Mandrichenko

March 29, 2006, Revised April 15, 2006

Preamble

There are two main motivations for the previous phase of evolution of the trigger database (TDB) under the TDB taskforce: first, to implement features required for the Version 15 trigger list for the IIb upgrade, and second to if possible modify the implementation to cut development time so that other desirable but not absolutely necessary features can be added with a reasonable effort. The TDB production release of March 20, 2006 met all the “Required” goals listed in the “D0 Trigger Database List of Requirements” (https://plone4.fnal.gov/P1/D0TDB/docs/D0TDB_requirements_final.pdf):

- C1) Trigger Transformation Tool
- N1a) Support for Splitting and Oring within single trigger list (an implementation based on null values for special scripts, used the original TDB schema unaltered)
- N2) Support for RunIIB tags

Also, significant progress has been made for two additional items:

- N3) Support for NOLUM exposure groups (Desirable)
- N5) Better support of trigsim users (Highly Desirable)

In addition, we removed the Corba-based Trigger DB server, which significantly simplified our design and made future modifications of the User Interface (UI) much easier. This also made possible better support for trigsim users.

As is not surprising, given the formation of the task force at the start of November 2005, not all desirable goals were met. In particular:

- The UI for entry of trigger lists is still rudimentary (An interface appropriate for entry of complex trigger lists was listed as Highly Desirable)
- Nolum testing was not completed in time for the production release

Also, an important result of our work is that now we have better understanding of Splitting/Oring - related functionality. In particular:

- It does not seem to be necessary to support mixed Splitting/Oring for a single L1 bit
- We now clearly see the need to implement the concept of an OR-group of L2 conditions

Other items had insufficient priority to be implemented by Feb 28, 2006 (the nominal start of shutdown).

Here we outline the next proposed steps for evolution of the TDB. Our highest priority (N1b) is to raise the level of support for complex trigger lists in the user interface. Next, a new requirement, support for association of TriggerNames to offline skims, has materialized; implementing this may address some other issues as well (N6). Finally, support for genuine L2Group names (N7) seems to address a number of the issues listed above, but requires a TDB schema change. The proposed plan proceeds in two stages, corresponding to production releases: UI improvements and skims in April and L2 Groups in May. We are assessing the feasibility of this schedule, which includes a first UI based on the current implementation of Splitting/Oring in the TDB, with a revised UI in the second release taking full advantage of L2Groups (which we are now committed to implementing). In addition, based on initial design of the V15 trigger list, we have found some ways requirements can be relaxed without seriously impacting the physics functionality of the TDB. Some items from the original requirements document are not part of this plan due to low priority, though simple aspects of some items might re-appear as maintenance requests. With the March release, we have in place sufficient infrastructure to begin entry of the V15 trigger list. The bulk of the work of entering a rather new list such as V15 lies in building the lower level objects. Most of the evolution of the TDB aims at simplifying the Triggermeister's tasks and addresses issues of entry the upper levels of the trigger list (building triggers from items at the script level or lower), and maintenance of these list.

Text in this font is material added to the original requirements document to describe the evolution plan. Text in Times New Roman has been extracted from the original requirements document. The new plans are given as modifications or additions to the original requirements, and we include comments on the status of some items from the original requirements document. We list the requirements and justification in the main text, and more details of the specification in appendices.

For definiteness, we repeat the prioritization information used in the initial requirements document, with the specific added information about the specific schedule for V15 trigger list development during the shutdown period.

Prioritization

Each item below is classified in several dimensions:

Time Scales

The time scale of implementation may directly affect the usefulness of the feature. Below is a time line for the continued V15 trigger list development from Marco Verzocchi.

End March	Start Entry of L1/L2 trigger list with full L3 structure and trigger names, but dummy L3 scripts.
15 April	L1/L2+dummy L3 entry complete
15-30 April	Enter full trigger list with real L3 scripts
15 May	Run full list on cosmic rays
1 June	Need SkimName support
1 June	Need L2Group name implemented
6 June	Data taking

Based on these considerations, we propose 2 main releases: April 10 and May 10

Priorities

- Required - essential
- Highly Desirable
- Desirable – implemented if time permits

Difficulty

This classification is mostly driven by the degree of structural change in database

- Straightforward - no structural changes to TDB; changes to xmlgen only, or entry of new items into database
- Moderate - addition of new attributes ("columns") to existing tables.
- Difficult - will require structural ("schema") changes to database, or substantial new code. In this document, schema changes mean new tables, or new relationships between tables.

Requirements for Evolution of the TDB

Present Functionality Completion

C1) Trigger Transformation Tool (TTT)

Rippling upward changes in L3 trigger elements but without changing shape of trigger tree.
Future evolution:

- □ Apply to a user-selectable part of a trigger list (not only the entire list).(Required, April, Moderate)
- □ Modification of multiple objects (Desirable, May, Moderate) The TTT requires visiting all objects above the modified object to change names/versions, which is time-consuming. Thus, it's desirable to allow changes to more than one object in a single session before having to ripple through these name changes. No use case to date, possibly because even one L3 change has been very hard. May be implemented under UI enhancements N1b instead.
- □ Allow not just replacement, but deletion and removal (as for L1 items). (Desirable; May, Moderate) Lower priority than modification of multiple objects.

Arnold Pompos will supply detailed specifications.

C2) Improve error detection and/or correction (Desirable, April, Moderate?)

C.2.a.2 Check for the use of objects with same name but different versions when building a trigger list; if found, warn the user after "Submit" but before a full commit. Detailed specs from Arnold.

C4) Maintenance

Bug fixes and smaller interface tweaks may be needed for the production-released code.

New trigger functionality

N1) L2 Splitting, Oring User Interface

N1b) (Highly Desirable), but for Evolution, priority raised to Required.
Splitting/Oring carries with it the implicit requirement for a user interface appropriate to support entry and maintenance of more complex trigger lists needed for Run IIb. As an example, this may entail handling of groups of related terms as a block. Not part of first production release. We are currently prioritizing components of this item.

Mixed Splitting and Oring within a single L1 bit is only used in one trigger in the draft V15 list. Therefore, this has been removed as a requirement and may be used to simplify implementation if convenient. Similarly, we have dropped a requirement for a "separator" between L1 triggers with differing L2 conditions, since the lack of splitting and Oring in the same trigger reduces the need for this functionality, and rearranging the trigger list provides an inelegant but effective workaround. Both of these may interact with a re-implementation of Splitting/Oring using L2GroupNames (N7), and bit counting (N1b.1)

N1b.1) User interface enhancements:

Entry Interface: (Required, April, Moderately Difficult)
 Manually selected groups: copy/paste by clicking (L1/L2/L3 elements)
 Prompt for how many L3 scripts you want for this L1/L2 pair condition

Entry Interface: (Very Highly Desirable, April, Straightforward)
 Text Entry of (at least) trigger lists by pasting TriggerNames/versions (built on copy/paste)

Report interface:
Counts for L2 bit displayed (Required, May, Moderately Difficult)
 Defer until L2Groups appear; main issue will be to note number of l2 bits consumed in the L2Or group; disappearance of nulls will simplify bit counting considerably.
 L3 bit counts are less important (Desirable), with real L2Groups for Or's, because this will supersede nulls and make the L3 bit count very close to the existing tl_index number

N1b.2) (Required, May, Moderate)
 L2 Or group names displayed by interface if necessary; manipulation by such groups
 Idea is to build manipulation interface first, based on current null implementation
 then replace these "pseudo" L2 groups by real, named L2 groups
 Unfortunately, these do not appear to be implementable by TriggerNameGroups, which depend on the TriggerName, not the L2ScriptNames
 Drop pseudo-groups names; use real L2Groups

N1b.3) (Required, May, Straightforward)
 The user interface should assist the triggermeister in maintaining the integrity of TriggerNames by forcing a new version or new name whenever an underlying component at L1, L2, or L3 is changed. Defer until real L2 groups appear.

N1b.4) (Desirable, May, Moderate)
 L3 group pseudo names: groups of L3 scripts (e.g. an Or Group) assigned a name to clarify that this group is identical to another one.
 Maybe more useful for users of trigger lists than for trigger meisters at entry (especially if lists are built by cutting and pasting).
 Possible implementations (could perhaps piggyback on implementation of L2 groups):
  Computed group name/number by interface, not database
   Assign a previous group number only if all scripts in this group
   identical to all scripts in previous group
  Full group implementation in database
  (probably not useful: the TriggerName is associated with a full triplet, not a L3 script)
  Hand assignment by TriggerGroupName mechanism
   (however, this does NOT enforce change when a component changes)

N1b.5) (Required, May, Straightforward)
 Skeleton report should report all relevant details; Display L2GroupNames
 Detailed specification by Arnold Pompos

N1b.6) (Highly Desirable, June 1, Straightforward) The report interface also needs to present information sufficient to understand the nolum exposure groups (see below).

N3) Support nolum (Highly Desirable, April 30, Straightforward)

Support for better physics use of exposure groups. Production candidate available for D0 review as of Jan 23, 2006. Not part of March production release because of lack of D0 testing, in part because the report interface needs some new features to allow users to assess the accuracy of the exposure groups.

- The report interface must match the xmlgen implementation of exposure groups generated using the nolum feature.
- Skeleton report possibly needs to include exposure group info.

N4) Parameters for L2 preprocessors (Required, April, Straightforward).

Currently, these are hard-coded in xmlgen, though they should be derived from the database. Not worked on for March production release due to other higher priority items. Drop database implementation; D0 (Arnold) to support the hard-coded xmlgen implementation for any RunIIb changes, with minimal CD advice. L2Cal group to provide new parameter names and values.

N6) TriggerGroupNames (TGN): (Required, April or May, Moderately Difficult)

Parameterized by TriggerGroups as the general concept, and having several Types (e.g. Skim Names and Analysis Tags). Associate a TriggerName (but not a specific version) to one or more TriggerGroups. This requires adding tables loosely coupled to trigger lists to the TDB. Note carefully that the TriggerNames live across trigger lists, and this is an association of all instances of a TriggerName, in all its versions, with a list of TGN's. Thus this association is NOT part of a triggerlist, but is independent of specific trigger lists. It may be necessary to enhance this feature to provide for reconstruction of the state of the names as of a particular date, or to have name/version controlled lists.

The primary use is for skims. There are many potential secondary uses, in which the TGN's are potentially overlapping analysis tags of various types. The meaning of analysis tags could be quite flexible, serving to organize triggerlists by particle content; by the group willing to "own" the bandwidth; by whether or not the TriggerName is prescaled; etc etc. Thus, this facility could be used to address items part of the requirement C3a. The analysis tags could also satisfy requirement N5b by providing the names used in the TriggerRateTool. Some thought should be devoted to whether this would require a third type (TRTTag) distinct from the generic analysis tags (perhaps with a naming convention to identify TRT tags). See Appendix for further details.

N7) L2GroupNames (Required, 1 June, Difficult)

L2 Group names in database. TriggerGroupNames cannot address all issues, because they are meant to be generic, cross trigger lists, not bind to specific versions of a TriggerName, and not be part of the trigger definition associated with a specific TriggerName. L2GroupNames are a more regularized representation of Splitting/Oring than the present null-based representation. The concept is to return to all triggers being represented by triplets, but generalizing the L2 element of the triplet to include multiple L2 scripts in the case of a L2 Or group. This implies changes in xmlgen, a translation mechanism to migrate triggers written with nulls into triggers written with groups at least semi-automatically, and report and entry interface changes.

Some specific advantages of this scheme, despite its greater implementation cost (which is why we had not attempted it for the February 28, 2006 deadline) are:

- All TriggerNames (not just some) are associated with an actual triplet of conditions
- It is straightforward to apply the name/version rule that any change in any of the members of the L2Group force a name (or version) change of the L2 group, which in turn forces a name or version change of the TriggerNames .
- L2GroupNames are a logical place to attach a friendlier name than the full script

See Appendix for further details.

The Appendix also contains a proposal for the contents of the two releases.

Appendices (Further Details of Some Items)

1. TriggerGroups (N6)

A table of a given type contains many to many associations of TriggerNames to TGNs of given type. The association is between a TriggerName, but is not specific to a particular version of that name. Similarly, the TGNs need not be versioned. The concept is an actively-maintained association represented in the table. Table entries look like:

TName1	GName3
TName1	GName7
TName20	GName2
TName20	GName3
TName3	GName44

Etc

This description of the table is the minimal representation. However, it should also be able to reconstruct the status of the table for a given time in the past. This most likely will mean having version control of the table of associations, so that committed changes force a new association version number and update the history information. The implementation may choose whether the table entries consist of names without versions, or specific names/versions (for example of the trigger names). In the latter case, by default a new version of a TriggerName should be associated with the sameTriggerGroupNames as the old version (this more-specific implementation contains details not necessarily of interest to users).

Constraints:

- All TNames in a (new) Global Triggerlist appear in at least one skim; (perhaps named “None”)
- Skims may be “abandoned” by not being associated with any TriggerName

We intend to use Oracle tools to export the skim structure to the Runs Database. This avoids adding skims to xml, and keeps the TDB free of access by the offline reconstruction. The Runs DB server and offline will be modified to access the skim tables.

User Interface:
Recognize I have entered a new TriggerName; offer choice of TGN Types, then select existing name(s) or add new ones for each selected TGN type.

The TGN-TriggerName association should also be maintainable outside the Entry interface, as the association might change even while a TriggerList remains fixed (e.g. it has already been used to take data, and is therefore frozen forever). The association changes if a new skim is defined, or skim contents rearranged. This interface could come somewhat later if convenient, as the entry interface could be used while constructing the trigger list, and the separate interface for fine-tuning.

Might want to look at lists ordered by TriggerNames and see all TGN’s associated with each, or vice versa, to help check the correctness of the association.

Perhaps need an option to select which TriggerNameTypes are listed in a report; a clickable link might give list of the TGN’s this TriggerName is associated with. It might be nice to have the ability to sort the triggers by tag.

Export-to-text facility for the analysis tag tables? D0-written python scripts are probably sufficient.

2. L2Groups (N7)

Contrast the current null-based representation of a Split/Or trigger:

L1	L2a	L3a	TNamea
L1	L2a	L3b	TNameb
L1	L2c	*Or	TNamec
L1	L2d	*Or	TNamed
L1	*Or	L3c	TNamee
L1	*Or	L3d	TNamef

with one using L2 groups: this becomes two L1 triggers, one pure splitting, and another pure Oring:

L1	L2a	L3a	TNamea
L1	L2a	L3b	TNameb

L1	L2Ga	L3c	TNamee
L1	L2Ga	L3d	TNamef

----

where the contents of the L2 Group L2Ga would be: {L2a, L2c, L2d} (since the L3 Or group {L3c, L3d} is to be applied to both split and or'd L2 triggers of the mixed trigger).

L2GroupNames should appear in the report and entry/manipulation interfaces where L2ScriptNames now do, with the L2 details just requiring a drill-down. Manipulations of the trigger list might assist formation of L2Groups. If the L2GroupNames are chosen to be simple enough, it will become easy to visually note reuse of groups.

It is straightforward to apply the name/version rule that any change in any of the members of the L2Group force a name (or version) change of the L2 group, which in turn forces a name or version change of the TriggerNames TNamee and Tnamef (that is any L3 bit affected).

It is a likely implementation choice that a new L2Group table appears with similar contents to, say, one of the ScriptName tables. Similarly, even Split triggers could have an L2Group (with a single entry) instead of an L2Script. In the example above, L2a could change to L2Gd, with L2Gd containing the single L2a script.

The fact that this scheme eliminates the need for nulls carries some other implications:

- The tl_index is again one to one with L3 bit number (no need for a separate L3 bit count)
- We dropped a requirement for a "separator" between L1 triggers with differing L2 conditions, since the lack of splitting and Oring in the same trigger reduces need for this functionality, and rearranging the trigger list provides an inelegant but effective workaround.

L3 group names done along L2Group lines are not as useful. This is primarily because even if a L3 group is re-used, the TriggerName must still be specified for each member of the L3 group. This name must be unique, because it is a friendly name for and an access method to the specific L3 bit set in the event when this L3 script succeeds. As such, it depends not only on the script content, but also on its context: which L1 and L2 conditions had to have been satisfied for this event to pass this filter. So, the need remains for a full triplet form, each with a unique TriggerName.

Recognizing reuse of L3 groups **is** useful for the physics user and triggermeister in constructing and understanding the trigger list, L3Groups might exist only in the interface (as computed group names or numbers) rather than as database tables.

3. Possible Contents of April and May Releases

Feature		Release
TTT selectable subset of trigger list		March (revised)
TTT multiple element replacement? Maybe=copy/paste;	if done	April/May
TTT removal/deletion?	if done	April/May
Cut/Copy/Paste selected items		April
Text entry of triggername/versions to Trigger List		April?
Notice object with inconsistent versions	if done	April?
L2 bit counts (with nulls)		Drop
Recognize when L2Or elt changes (nulls)		Drop
L2 Group recognition/computed names (nulls)		Drop
Report info for nolum (and maybe skeleton report)		April
Nolum		April
L2Cal parameters from TDB		Drop
L2Cal parameters: xlmgen hardcoded		April
skim names/TriggerGroupNames		April/May as convenient
analysis tags		April/May as convenient
TriggerRateTool tags ?	if done	April/May as convenient
skim table versioning		April/May as convenient
skim table revision interface		April/May as convenient
skim table export to Runs DB		April/May as convenient
L2Groups		May
Tool to port trigger list with nulls to L2group lists		May
L2 bit counts (with L2 Groups)		May
Recognize when L2 or elt changes		May (with L2Groups)
L2 Group recognition		May (with L2Groups)
L3 group names (probably computed)?	if done	May???
Skeleton report (L2groups)		May